post_incompact3d

post_incompact3d is an unofficial post-processing program firstly developed by R. Corsini (University of Modena and Reggio Emilia, UNIMORE) based on Incompact3d version 2.0. This new version is based on Incompact3d v4.0, and it is able to perform calculations of flow statistics with 2 homogeneous directions (x and z), with different flow realizations and (if requested) with averages in time. It is assumed that the `2decomp-fft` library has been installed. For details, see the 'Quick Incompact3d guide and notes'.

Compiling options

The compilation is made with `cmake`, that needs to be run in a *build* directory. The default compilation is performed with the option `TTBL_MODE=ON`, that ensures statistics with averages along homogeneous directions x and z and with different flow realizations. Statistics are printed for each time-unit of snapshots' saving (this is used for temporal cases, e.g. Temporal Turbulent Boundary Layers (TTBLs) and more in general for temporal simulations, e.g. temporal jets). With the default compilation, the program is in *TTBL Mode*.

Default compilation (TTBL Mode)

```
cmake ../
```

Optionally, it is possible to compile the program with the option `TTBL_MODE=OFF`. This option allows to average the snapshots also in time (this is used for statistically steady cases, e.g. channel flows). With this option disabled, the program is in *Channel mode*.

Optional compilation (Channel Mode)

```
cmake -DTTBL_MODE=OFF ../
```

The specific compiling option can be also modified by changing the value `TTBL_MODE:BOOL` inside the `CMakeCache.txt` file that is present inside the `build` directory if `cmake` has been run.

Finally, it is also possible to impose a specific compiler (e.g. `ifort`, in *Channel Mode*):

```
cmake -DTTBL_MODE=OFF -DCMAKE_Fortran_COMPILER=ifort ../
```

and equivalently for *TTBL Mode*:

```
cmake -DTTBL_MODE=ON -DCMAKE_Fortran_COMPILER=ifort ../
```

Usage

`post_incompact3d` must be run inside the folder of a specific flow case. If different flow realizations are present, the different realizations must be stored in different *data* folders, that must be named `data_rn` where `n` is the number of the specific flow realization. If only one realization is present, no modifications are required to the standard folder name `data`.

`post_incompact3d` requires the file `post.prm` as input file. In this file, the following parameters must be specified:

- `flowcasename`: flow case name added to the images' filename generated with `plot_statistics.py`
- `file1`: first snapshot index (index of the snapshot's name)
- `filen`: final snapshot index (index of the snapshot's name)
- `icrfile`: file increment (index increment)
- `nr`: number of flow realizations

- `post_mean`: compute mean statistics (0: no, 1: yes)
- `post_vort`: compute mean vorticity and mean gradients (0: no, 1: yes)
- `post_diss`: compute mean total dissipation rate (0: no, 1: yes)
- `post_corz`: compute correlation functions along z (0: no, 1: yes) (mean statistics must be already calculated).
- `post_tke_eq`: compute Turbulent Kinetic Energy (TKE) budgets (0: no, 1: yes) (mean statistics must be already calculated).

Statistics
As following, we collect the flow statistics calculated.

- **Velocity field (O(6))**
  - Averages
    $$\langle u \rangle, \langle v \rangle, \langle w \rangle$$
  - Variances
    $$\langle u'^2 \rangle, \langle v'^2 \rangle, \langle w'^2 \rangle$$
  - Skewnesses
    $$\mathsf{skew}[u], \mathsf{skew}[v], \mathsf{skew}[w]$$
  - Kurtoses
    $$\mathsf{kurt}[u], \mathsf{kurt}[v], \mathsf{kurt}[w]$$
- **Reynolds stresses (O(6))**
  $$\langle u'v' \rangle, \langle u'w' \rangle, \langle v'w' \rangle$$
- **Pressure field (O(6))**
  - Average and variance
    $$\langle p \rangle, \langle p'^2 \rangle$$
- **Scalar field (O(6))**
  - Average and variance
    $$\langle \varphi \rangle, \langle \varphi'^2 \rangle$$
  - Mixed fluctuations
    $$\langle u'\varphi' \rangle, \langle v'\varphi' \rangle, \langle w'\varphi' \rangle$$
- **Vorticity field (O(6))**
  - Averages
    $$\langle \omega_x \rangle, \langle \omega_y \rangle, \langle \omega_z \rangle$$
- **Mean gradients (O(6))**
  - Mean streamwise gradient
    $$\langle \frac{\partial u}{\partial y} \rangle = \frac{\partial U}{\partial y}$$
  - Mean spanwise gradient

$$\langle \frac{\partial w}{\partial y} \rangle = \frac{\partial W}{\partial y}$$

- **Mean scalar gradient**
  - Mean scalar gradient

$$\langle \frac{\partial \varphi}{\partial y} \rangle = \frac{\partial \Phi}{\partial y}$$

- **Total dissipation rate (O(6))**
  - Average

$$\langle \varepsilon \rangle = 2\nu \langle S_{ij} S_{ij} \rangle$$

    where double contraction on the indexes i and j is performed.

- **Correlation functions**
  A previous run must be performed in order to calculate the averages necessary to calculate the needed fluctuations (both for *Channel* and *TTBL Mode*).
  - Auto-correlation functions for fluctuations of velocity components in spanwise direction (z):

$$R_{ii}(r_z) = \langle u_i'(x, y, z + r_z, t) u_i'(x, y, z, t) \rangle$$

    where index i stands for the three directions x, y and z.
  - Correlation function in spanwise direction (z) for mixed fluctuations u' and v':

$$R_{uv}(r_z) = \langle u'(x, y, z + r_z, t) v'(x, y, z, t) \rangle$$

  - Auto-correlation function for scalar field fluctuations:

$$R_{\varphi\varphi}(r_z) = \langle \varphi'(x, y, z + r_z, t) \varphi'(x, y, z, t) \rangle$$

- **Turbulent kinetic energy budgets**
  The turbulent kinetic energy equation is computed in the following form (due to the statistical homogeneity in x and z):

$$\frac{\partial}{\partial y} \langle k v' \rangle + \frac{1}{\rho} \frac{\partial}{\partial y} \langle p' v' \rangle - \nu \frac{\partial^2}{\partial y^2} \langle k \rangle = - \langle u' v' \rangle \frac{\partial U}{\partial y} - \nu \langle \frac{\partial u_i'}{\partial x_j} \frac{\partial u_i'}{\partial x_j} \rangle$$

  where $k = (1/2) u_i' u_i'$ is the turbulent kinetic energy.

Statistics involving calculations of integrals

For integral quantities of TTBL simulations (e.g displacement thickness δ^*, momentum thickness θ, etc.), the user can employ the python function `ttbl_indexes_snap.py` that can be found in the same parent directory. This function employs the post-processing data generated with `post_incompact3d`, so we are calculating these quantities only at the time instant corresponding to snapshots' savings. For a full time evolution, use `cf_monitoring`.